09/769,956

L Number	Hits	Search Text	DB	Time stamp
-	2	("6397352").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 13:59
-	125	(error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:57
-	1004373	monitor$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:03
-	13576	memory adj bus$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:04
-	12329	READ adj command	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:05
-	0	((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) with monitor$4 with (memory adj bus$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:05
-	3	(((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) and monitor$4 and (memory adj bus$4)) and (READ adj command)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:05
-	25	((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) and monitor$4 and (memory adj bus$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:16
-	21138	detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:50
-	45	((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) with (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:50

-	3299	host adj controller	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:51
-	7	(((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) with (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (host adj controller)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:54
-	60	((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) same (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:54
-	11	(((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) same (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (READ adj command)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:55
-	4	((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) same (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (READ adj command)) not (((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) with (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (host adj controller)) or (((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) and monitor$4 and (memory adj bus$4)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:55
-	13	(error$4 or fault$4 or fail$4 or problem$) with cleans$4 with memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 14:58

-	8	((error$4 or fault$4 or fail$4 or problem$) with cleans$4 with memory) not ((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) and monitor$4 and (memory adj bus$4)) or ((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) with (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (host adj controller)) or ((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) same (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (READ adj command)) or (((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) same (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (READ adj command)) not (((((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) with (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)) and (host adj controller)) or (((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory) and monitor$4 and (memory adj bus$4)))))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:00
-	3539	(714/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:00
-	2234	(711/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:00
-	1939	(365/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:00
-	7638	((714/?).ccls.) or ((711/?).ccls.) or ((365/?).ccls.)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:00
-	30	(((714/?).ccls.) or ((711/?).ccls.) or ((365/?).ccls.)) and ((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:00

-	27	((((714/?).ccls.) or ((711/?).ccls.) or ((365/?).ccls.)) and ((error$4 or fault$4 or fail$4 or problem$) with (scrub$4 or cleans$4) with memory)) and (detect$4 with (error$4 or fault$4 or fail$4 or problem$) with memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/10 15:01

09/769,956

L Number	Hits	Search Text	DB	Time stamp
1	249	(error$4 or fault$4 or fail$4 or problem$) same (cleans$4 or scrub$4) same memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 10:54
2	28300	data adj word$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 10:54
3	530425	detect$4 same (error$4 or fault$4 or fail$4 or problem$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 10:55
4	156925	monitor$4 same (error$4 or fault$4 or fail$4 or problem$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 10:56
5	25	((error$4 or fault$4 or fail$4 or problem$) same (cleans$4 or scrub$4) same memory) and (data adj word$) and (detect$4 same (error$4 or fault$4 or fail$4 or problem$)) and (monitor$4 same (error$4 or fault$4 or fail$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:06
6	1	dynamic$4 adj schedul$4 adj access	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:08
7	18978	(READ or WRITE) adj command$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:08
8	13	(((error$4 or fault$4 or fail$4 or problem$) same (cleans$4 or scrub$4) same memory) and (data adj word$) and (detect$4 same (error$4 or fault$4 or fail$4 or problem$)) and (monitor$4 same (error$4 or fault$4 or fail$4 or problem$))) and ((READ or WRITE) adj command$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:09
9	4752	scrub$4 and cleans$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:10

10	1465	scrub$4 with cleans$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:10
11	1	(scrub$4 with cleans$4) and (data adj word$) and (detect$4 same (error$4 or fault$4 or fail$4 or problem$)) and (monitor$4 same (error$4 or fault$4 or fail$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:11
12	1	(scrub$4 with cleans$4) and (data adj word$) and (detect$4 same (error$4 or fault$4 or fail$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:11
13	1	(scrub$4 with cleans$4) and (data adj word$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:11
14	2	(scrub$4 and cleans$4) and (data adj word$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:12
15	288	(scrub$4 and cleans$4) and memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:12
16	2	((scrub$4 and cleans$4) and memory) and (data adj word$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:12
17	3542	(714/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:12
18	2235	(711/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:12
19	1939	(365/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:13

20	7642	((714/?).ccls.) or ((711/?).ccls.) or ((365/?).ccls.)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:13
21	4	(((714/?).ccls.) or ((711/?).ccls.) or ((365/?).ccls.)) and ((scrub$4 and cleans$4) and memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/11 11:13
22	1		USPAT	2003/09/11 11:18
23	1		USPAT	2003/09/11 11:18
24	1		USPAT	2003/09/11 11:18
25	1		USPAT	2003/09/11 11:18
26	1		USPAT	2003/09/11 11:19
27	1		USPAT	2003/09/11 11:19
28	1		USPAT	2003/09/11 11:19
29	1		USPAT	2003/09/11 11:19
30	1		USPAT	2003/09/11 11:19
31	1		USPAT	2003/09/11 11:19
32	1		USPAT	2003/09/11 11:21
33	1		USPAT	2003/09/11 11:21
34	1		USPAT	2003/09/11 11:22
35	1		USPAT	2003/09/11 11:22
36	1		USPAT	2003/09/11 11:22
37	1		USPAT	2003/09/11 11:22
38	1		USPAT	2003/09/11 11:22
39	1		USPAT	2003/09/11 11:22
40	1		USPAT	2003/09/11 11:23
41	1		USPAT	2003/09/11 11:23

An alternative data integrity maintenance system is disclosed in U.S. Pat. No. 5,210,860 which discloses an intelligent disk array controller. This controller activates a data read subroutine during periods of no data read/write activity to sequentially read each **memory** location in the disk array **memory**. The read operation determines whether each **memory** location can be read, and any detected errors are corrected as they are identified. This operation sequentially cycles through all **memory** locations on all disk drives in the system. A problem with this system is that recently written **memory** locations are not read until the system cycles through its predetermined sequence of disk drives. In addition, if there is a significant amount of data read/write activity, this single verification process can take a substantial amount of time before verifying the data written on the disk drives. Furthermore, unused portions of **memory** are checked as part of the sequence, such as spare drives. This process is therefore an improvement but suffers from a number of performance impediments.

Brief Summary Text - BSTX (9):

The above described problems are solved and a technical advance achieved in the field by the disk **scrubbing** system for a data storage subsystem. This system avoids the data integrity problems of the prior art by periodically verifying the integrity of the data stored on the disk drives of the data storage subsystem. This is accomplished by one or more background processes that cycle through predetermined segments of active **memory** to verify the integrity of the data stored therein. A priority **scrubbing** queue is also available to note data storage locations that have recently had data written thereon by the host processor and which require a more timely review of the data than the data storage locations that have not had data written therein since the last periodic data **scrubbing** operation.

Brief Summary Text - BSTX (10):

The disk drives in a disk drive array data storage subsystem are configured into a plurality of variable size redundancy groups of N+M parallel connected

disk drives to store data thereon. The disk drive array data storage subsystem
dynamically maps between three abstract layers: virtual, logical and physical.
The virtual layer functions as a conventional large form factor disk drive
memory. The logical layer functions as an array of storage units that are
grouped into a plurality of redundancy groups, each containing N+M physical
disk drives. The physical layer functions as a plurality of individual small
form factor disk drives. A controller in the data storage subsystem operates
to effectuate the dynamic mapping of data among these abstract layers and to
control the allocation and management of the actual space on the physical
devices. These data storage management functions are performed in a manner
that renders the operation of the disk drive array data storage subsystem
transparent to the host processor which perceives only the virtual image of the
data storage subsystem.

Brief Summary Text - BSTX (11):

When data is written to available **memory** space on a disk drive in a redundancy group, the physical tracks on which the data is stored are grouped
together into logical cylinders and are noted in the logical cylinder table as
containing newly-written data. A priority **scrub** routine sequences through all
newly written physical tracks in the logical cylinder to verify the integrity
of the data stored in the physical tracks by performing a data readback and
error check operation. The priority **scrub** routine operates to perform a timely
read and verify after write operation to detect and correct errors created
during the data write process. A plurality of concurrently operational
periodic disk **scrub** routines periodically sequences through all physical tracks
in the data storage subsystem which contain customer or redundancy data to
perform a data readback and error check operation on these physical tracks.
The active data stored in the data storage subsystem is thereby routinely
checked to ensure the integrity of this data.

Drawing Description Text - DRTX (6):

FIG. 5 illustrates the **memory** space of the data storage subsystem as viewed

by the disk **scrub** operations;

Drawing Description Text - DRTX (9):
FIG. 9 illustrates in flow diagram form the operation of the periodic disk **scrubbing** operation;

Drawing Description Text - DRTX (10):
FIG. 10 illustrates in flow diagram form the operation of the logical cylinder **scrubbing** operation;

Drawing Description Text - DRTX (11):
FIG. 11 illustrates in flow diagram form the operation of the periodic **scrub** rate adjustment operation;

Drawing Description Text - DRTX (12):
FIG. 12 illustrates in flow diagram form the operation of the track **scrubbing** operation; and

Drawing Description Text - DRTX (13):
FIG. 13 illustrates in flow diagram form the operation of the priority disk **scrubbing** operation.

Detailed Description Text - DETX (2):
The data storage subsystem of the present invention uses a plurality of small form factor disk drives in place of a single large form factor disk drive to implement an inexpensive, high performance, high reliability disk drive **memory** that emulates the format and capability of large form factor disk drives. The plurality of disk drives in the disk drive array data storage subsystem are configured into a plurality of variable size redundancy groups of N+M connected disk drives to store data thereon. Each redundancy group, also called a logical disk drive, is divided into a number of logical cylinders, each containing i logical tracks, one logical track for each of the i physical tracks contained in a cylinder of one physical disk drive. Each logical track is comprised of N+M physical tracks, one physical track from each disk drive in the redundancy group. The N+M disk drives are used to store N data segments, one on each of N physical tracks per logical track, and to store M

redundancy segments, one on each of M physical tracks per logical track in the redundancy group. The N+M disk drives in a redundancy group have unsynchronized spindles and loosely coupled actuators. The data is transferred to the disk drives via independent reads and writes since all disk drives operate independently. Furthermore, the M redundancy segments, for successive logical cylinders, are distributed across all the disk drives in the redundancy group rather than using dedicated redundancy disk drives.

Detailed Description Text - DETX (3):

The disk drive array data storage subsystem includes a controller that dynamically maps between three abstract layers: virtual, logical and physical. The virtual layer functions as a conventional large form factor disk drive **memory**. The logical layer functions as an array of storage units that are grouped into a plurality of redundancy groups, each containing N+M physical disk drives. The physical layer functions as a plurality of individual small form factor fixed block architecture (FBA) disk drives. The controller effectuates the dynamic mapping of data among these abstract layers and controls the allocation and management of the actual space on the physical devices. These data storage management functions are performed in a manner that renders the operation of the data storage subsystem transparent to the host processor, which perceives only the virtual image of the data storage subsystem.

Detailed Description Text - DETX (4):

The performance of this system is enhanced by the use of a cache **memory** with both volatile and non-volatile portions and "backend" data staging and destaging processes. Data received from the host processors is stored in the cache **memory** in the form of modifications to data already stored in the redundancy groups of the data storage subsystem. No data stored in a redundancy group is modified. A virtual track is staged from a redundancy group into cache. The host then modifies some, perhaps all, of the records on the virtual track. Then, as determined by cache replacement algorithms, the modified virtual track is selected to be destaged to a redundancy group. When

thus selected, a virtual track is divided (marked off) into several physical sectors to be stored on one or more physical tracks of one or more logical tracks. A complete physical track may contain physical sectors from one or more virtual tracks. Each physical track is combined with N-1 other physical tracks to form the N data segments of a logical track.

Detailed Description Text - DETX (5):
The original, unmodified data that is still stored in a redundancy group is simply flagged as obsolete. Obviously, as data is modified, the redundancy groups increasingly contain numerous virtual tracks of obsolete data. The remaining valid virtual tracks in a logical cylinder are read to the cache **memory** in a background "free space collection" process. They are then written to a previously emptied logical cylinder and the "collected" logical cylinder is tagged as being empty. Thus, all redundancy data creation, writing and free space collection occurs in background, rather than as on-demand processes.
This arrangement avoids the parity update problem of existing disk drive array systems and improves the response time versus access rate performance of the data storage subsystem by transferring these overhead tasks to background processes.

Detailed Description Text - DETX (9):
Control unit 101 includes two cluster controls 111, 112 for redundancy purposes. Within a cluster control 111 the multipath storage director 110-0 provides a hardware interface to interconnect data channels 21, 31 to cluster control 111 contained in control unit 101. In this respect, the multipath storage director 110-0 provides a hardware interface to the associated data channels 21, 31 and provides a multiplex function to enable any attached data channel (for example 21) from any host processor (for example 11) to interconnect to a selected cluster control 111 within control unit 101. The cluster control 111 itself provides a pair of storage paths 200-0, 200-1 which function as an interface to a plurality of optical fiber backend channels 104.
In addition, the cluster control 111 includes a data compression

function as well as a data routing function that enables cluster control 111 to direct the transfer of data between a selected data channel 21 and cache **memory** 113, and between cache **memory** 113 and one of the connected optical fiber backend channels 104. Control unit 101 provides the major data storage subsystem control functions that include the creation and regulation of data redundancy groups, reconstruction of data for a failed disk drive, switching a spare disk drive in place of a failed disk drive, data redundancy generation, logical device space management, and virtual to logical device mapping.

Detailed Description Text - DETX (12):

FIG. 2 illustrates in block diagram form additional details of cluster control 111. Multipath storage director 110 includes a plurality of channel interface units 201-0 to 201-7, each of which terminates a corresponding pair of data channels 21, 31. The control and data signals received by the corresponding channel interface unit 201-0 are output on either of the corresponding control and data buses 206-C, 206-D, or 207-C, 207-D, respectively, to either storage path 200-0 or storage path 200-1. Thus, as can be seen from the structure of the cluster control 111 illustrated in FIG. 2, there is a significant amount of symmetry contained therein. Storage path 200-0 is identical to storage path 200-1 and only one of these is described herein. The multipath storage director 110 uses two sets of data and control busses 206-D, C and 207-D, C to interconnect each channel interface unit 201-0 to 201-7 with both storage path 200-0 and 200-1 so that the corresponding data channel 21 from the associated host processor 11 can be switched via either storage path 200-0 or 200-1 to the plurality of optical fiber backend channels 104. Within storage path 200-0 is contained a processor 204-0 that regulates the operation of storage path 200-0. In addition, an optical device interface 205-0 is provided to convert between the optical fiber signalling format of optical fiber backend channels 104 and the metallic conductors contained within storage path 200-0. Channel interface control 202-0 operates under control of processor 204-0 to control the flow of data to and from cache **memory** 113 and one of the channel interface units 201 that is presently active with

storage path 200-0. The channel interface control 202-0 includes a cyclic redundancy check (CRC) generator/checker to generate and check the CRC bytes for the received data. The channel interface circuit 202-0 also includes a buffer that compensates for speed mismatch between the data transmission rate of the data channel 21 and the available data transfer capability of the cache **memory** 113. The data that is received by the channel interface control circuit 202-0 from a corresponding channel interface circuit 201 is forwarded to the cache **memory** 113 via channel data compression circuit 203-0. The channel data compression circuit 203-0 provides the necessary hardware and microcode to perform compression of the channel data for the control unit 101 on a data write from the host processor 11. It also performs the necessary decompression operation for control unit 101 on a data read operation by the host processor 11.

Detailed Description Text - DETX (13):

As can be seen from the architecture illustrated in FIG. 2, all data transfers between a host processor 11 and a redundancy group in the disk drive subsets 103 are routed through cache **memory** 113. Control of cache **memory** 113 is provided in control unit 101 by processor 204-0. The functions provided by processor 204-0 include initialization of the cache directory and other cache data structures, cache directory searching and management, cache space management, cache performance improvement algorithms as well as other cache control functions. In addition, processor 204-0 creates the redundancy groups from the disk drives in disk drive subsets 103 and maintains records of the status of those devices. Processor 204-0 also causes the redundancy data across the N data disks in a redundancy group to be generated within cache **memory** 113 and writes the M segments of redundancy data onto the M redundancy disks in the redundancy group. The functional software in processor 204-0 also manages the mappings from virtual to logical and from logical to physical devices. The tables that describe this mapping are updated, maintained, backed up and occasionally recovered by this functional software on processor 204-0. The free space collection function is also performed by processor 204-0 as well

as management and scheduling of the optical fiber backend channels 104. Many of these above functions are well known in the data processing art and are not described in any detail herein.

Detailed Description Text - DETX (17):

With respect to data transfer operations, all data transfers go through cache **memory** 113. Therefore, front end or channel transfer operations are completely independent of backend or device transfer operations. In this system, staging operations are similar to staging in other cached disk subsystems but destaging transfers are collected into groups for bulk transfers. In addition, this data storage subsystem 100 simultaneously performs free space collection, mapping table backup, and error recovery as background processes. Because of the complete front end/backend separation, the data storage subsystem 100 is liberated from the exacting processor timing dependencies of previous count key data disk subsystems. The subsystem is free to dedicate its processing resources to increasing performance through more intelligent scheduling and data transfer control.

Detailed Description Text - DETX (18):

When the host processor 11 transmits data over the data channel 21 to the data storage subsystem 100, the data is transmitted in the form of the individual records of a virtual track. In order to render the operation of the disk drive array data storage subsystem 100 transparent to the host processor 11, the received data is stored on the actual physical disk drives (122-1 to 122-n+m) in the form of virtual track instances which reflect the capacity of a track on the large form factor disk drive that is emulated by data storage subsystem 100. Although a virtual track instance may spill over from one physical track to the next physical track, a virtual track instance is not permitted to spill over from one logical cylinder to another. This is done in order to simplify the management of the **memory** space.

Detailed Description Text - DETX (21):

It is necessary to accurately record the location of all data within the disk drive array data storage subsystem 100 since the data received from the

host processors 11-12 is mapped from its address in the virtual space to a physical location in the subsystem in a dynamic fashion. A virtual track directory is maintained to recall the location of the present instance of each virtual track in disk drive array data storage subsystem 100. Changes to the virtual track directory are journaled to a non-volatile store and are backed up with fuzzy image copies to safeguard the mapping data. The virtual track directory 4 consists of an entry 400 (FIG. 4) for each virtual track which the associated host processor 11 can address. The virtual track directory entry 400 also contains data 407 indicative of the length of the virtual track instance in sectors. The virtual track directory 4 is stored in noncontiguous pieces of the cache **memory** 113 and is addressed indirectly through pointers in a virtual device table. The virtual track directory 4 is updated whenever a new virtual track instance is written to the disk drives.

Detailed Description Text - DETX (22):

The storage control also includes a free space directory 800 (FIG. 8) which is a list of all of the logical cylinders in the disk drive array data storage subsystem 100 ordered by logical device. Each logical device is cataloged in a list called a free space list 801 for the logical device; each list entry represents a logical cylinder and indicates the amount of free space that this logical cylinder presently contains. This free space directory contains a positional entry for each logical cylinder; each entry includes both forward 802 and backward 803 pointers for the doubly linked free space list 801 for its logical device and the number of free sectors contained in the logical cylinder. Each of these pointers 802, 803 points either to another entry in the free space list 801 for its logical device or is null. In addition to the pointers and free sector count, the free space directory also contains entries that do not relate to free space, but relate to the logical cylinder. There is a flag byte known as the logical cylinder table (LCT) which contains, among other flags, a C flag and some T flags. The C flag indicates that the logical cylinder has been written to and requires priority **scrubbing**. The T

flags indicate states of the logical cylinder when the logical cylinder should not be **scrubbed,** such as logical cylinder is being written, logical cylinder is being free space collected, and logical cylinder is being reconstructed. The collection of free space is a background process that is implemented in the disk drive array data storage subsystem 100. The free space collection process makes use of the logical cylinder directory, which is a list contained in the last few sectors of each logical cylinder indicative of the contents of that logical cylinder. The logical cylinder directory contains an entry for each virtual track instance contained within the logical cylinder. The entry for each virtual track instance contains the identifier of the virtual track instance and the identifier of the relative sector within the logical cylinder in which the virtual track instance begins. From this directory and the virtual track directory, the free space collection process can determine which virtual track instances are still current in this logical cylinder and therefore need to be moved to another location to make the logical cylinder available for writing new data.

Detailed Description Text - DETX (24):

FIG. 6 illustrates in flow diagram form the operational steps taken by processor 204 in control unit 101 of the data storage subsystem 100 to read data from a data redundancy group 122-1 to 122-n+m in the disk drive subsets 103. The disk drive array data storage subsystem 100 supports reads of any size. However, the logical layer only supports reads of virtual track instances. In order to perform a read operation, the virtual track instance that contains the data to be read is staged from the logical layer into the cache **memory** 113. The data record is then transferred from the cache **memory** 113 and any clean up is performed to complete the read operation.

Detailed Description Text - DETX (25):

At step 601, the control unit 101 prepares to read a record from a virtual track. At step 602, the control unit 101 branches to the cache directory search subroutine to assure that the virtual track is located in the cache

memory 113 since the virtual track may already have been staged into the cache
memory 113 and stored therein in addition to having a copy stored on the
plurality of disk drives (122-1 to 122-n+m) that constitute the redundancy
group in which the virtual track is stored. At step 603, the control unit 101
scans the hash table directory of the cache **memory** 113 to determine whether the
requested virtual track is located in the cache **memory** 113. If it is, at step
604 control returns back to the main read operation routine and the cache
staging subroutine that constitutes steps 605-616 is terminated.

Detailed Description Text - DETX (26):

Assume, for the purpose of this description, that the virtual track that has
been requested is not located in the cache **memory** 113. Processing proceeds to
step 605 where the control unit 101 looks up the address of the virtual track
in the virtual to logical map table. At step 606, the logical map location is
used to map the logical device to one or more physical devices in the
redundancy group. At step 607, the control unit 101 schedules one or more
physical read operations to retrieve the virtual track instance from
appropriate ones of identified physical devices 122-1 to 122-n+m. At step 608,
the control unit 101 clears errors for these operations. At step 609, a
determination is made whether all the reads have been completed, since the
requested virtual track instance may be stored on more than one of the N+M disk
drives in a redundancy group. If all of the reads have not been completed,
processing proceeds to step 614 where the control unit 101 waits for the next
completion of a read operation by one of the N+M disk drives in the redundancy
group. At step 615 the next reading disk drive has completed its operation and
a determination is made whether there are any errors in the read operation that
has just been completed. If there are errors, at step 616 the errors are
marked and control proceeds back to the beginning of step 609 where a
determination is made whether all the reads have been completed. If at this
point all the reads have been completed and all portions of the virtual track
instance have been retrieved from the redundancy group, then processing
proceeds to step 610 where a determination is made whether there are any errors